Exploring Major and Minor Clusters in Massive Databases (The Numerical Solution of Differential Equations and Linear Computation)

AUTHOR(S):

Kobayashi, Mei; Aono, Masaki

数理解析研究所講究録 1320 巻 2003 年 250-260

大規模データベースの主要クラスタ・二次クラスタの探索

小林 メイ, 青野 雅樹 (日本 IBM・東京基礎研)
〒 242-8502 神奈川県大和市下鶴間 1623-14

Exploring Major and Minor Clusters in Massive Databases

Mei Kobayashi and Masaki Aono
{mei, aono}@jp.ibm.com , IBM Research, Tokyo Research Laboratory
1623-14, Shimotsuruma, Yamato-shi, Kanagawa-ken 242-8502 Japan

We present a novel search and cluster mining system based on vector space modeling that addresses some issues that have been neglected by conventional systems for database analysis. Novel features of our search and cluster mining engine are: discovery of both major and minor clusters, accommodation of cluster overlap, automatic labeling of clusters based on their document contents, and advanced visualization of search and mining results. Implementation studies using a data set with over 100,000 news articles demonstrate the effectiveness of our system.

1. Introduction

The proliferation of massive databases has created unforeseen challenges for many enterprises. One of these challenges is to develop tools for analyzing massive repositories of heterogeneously formatted documents, generated by many people and machines. Some successful methods for retrieving and analyzing information have been developed by the data mining community[1], however, there is significant room for improvement. We present a novel system for exploring the contents of massive databases that improves upon conventional mining systems in several ways.

First, most data mining systems consider numerical data of homogeneous format and target major cluster discovery and analysis, even though major topics are often already well-known by personnel from on-the-job experience. However, information on minor clusters is usually not known, and until recently, their discovery and analysis have been neglected, although it may be just as valuable for business and government planning [8]. For example, minor clusters in a customer survey database may represent emerging trends or long-term, minor concerns that may lead to customer dissatisfaction. Or minor clusters may represent loyal, so-called *gold* customers or very bad customers who may default on a loan. In scientific databases, minor clusters may aid in the accurate diagnosis of diseases or prediction of natural disasters.

Second, whereas most conventional clustering systems are partition-based, ours does not partition the document set. We recognize that cluster overlap is a naturally occurring phenomenon in very large databases. Moreover, preservation of overlap information is essential for analysis of database contents and preservation of essential documents [4]; portions of very small clusters that have substantial overlap with other clusters might be trimmed so much by partitioning

[1] *www.kdnuggets.com*

algorithms that the tiny fraction of the cluster that remains may be mistaken for noise and discarded. Finally, very few mining systems have an advanced graphical user interface (GUI) and system to aid selection of good coordinate axes and angles in 3-D space for visualizing results.

In this paper we present a novel cluster mining system with an advanced GUI to aid in the understanding of contents of massive databases. The system finds and automatically labels both major and minor clusters. The remainder of this paper is organized as follows. In the next section we review work related to ours. The third section is a description of the main components of our system. Results from implementation studies using a very large set of over 100,000 news articles from the TREC benchmark set[2] are given in the penultimate section. We conclude with a summary of findings and discuss possible directions for future research.

2. Related Works

Vector space modeling (VSM) of databases has become a standard tool in search and clustering systems since its introduction by Salton over three decades ago [2],[9]. One of the advantages of the method is it enables relevance ranking of documents of heterogeneous format with respect to user input queries as long as the attributes are well-defined characteristics of the documents. In *Boolean* vector models each coordinate of the vector is naught (when the corresponding attribute is absent) or unity (when the corresponding attribute is present). In our implementation studies we used a fairly common type of *term frequency inverse document frequency weighting (tf-idf)* to take into account the frequency of their appearance in each document as well as in the document set as a whole. The weight of the i-th term in the j-th document, denoted by weight(i,j) is defined by:

$$\text{weight}(i,j) = \begin{cases} (1+tf_{i,j})\log_2(N/df_i), & \text{if } tf_{i,j} \geq 1\,, \\ 0\,, & \text{if } tf_{i,j} = 0\,, \end{cases}$$

where $tf_{i,j}$ is defined as the number of occurrences of the i-th term within the j-th document d_j, and df_i is the number of documents in which the term appears. Each query is modeled as a vector using the same attribute space as the documents. The relevancy ranking of a document with respect to a query depends on its "*distance*" to the query vector. In our experiments we use the cosine of the angle defined by the query and document vectors as the distance metric.

Many databases are so massive that the similarity ranking method described above requires too many computations and comparisons for real-time output[3]. One approach to solving this problem is to reduce the dimension of mathematical models by projecting into a subspace of sufficiently small dimension to enable fast response times, but large enough to retain characteristics for distinguishing contents of individual documents. Two algorithms for carrying out dimensional reduction are: *latent semantic indexing* (LSI) [3], and a variation of principal component analysis (PCA), which we refer to hereafter as *covariance matrix analysis* or *COV* [7].

[2]U.S. National Institute of Standards & Technology Text REtrieval Competition: *http://trec.nist.gov*

[3]Scalability of relevancy ranking methods to massive databases is a serious concern as users consistently select the most important feature of IR engines to be fast, real-time response to their queries in the annual Graphics, Visualization, and Usability Center of Georgia Institute of Technology's Web users' survey: *http://www.gvu.gaetch.edu/user_surveys.*

Let A be the M-by-N document-attribute matrix representing a database with M documents modeled by N attributes, with entry $a(i,j)$ representing the importance of the i-th term in the j-th document. The main idea in LSI is to reduce the dimension of the IR problem to k, where $k \ll M, N$, by projecting the problem into the space spanned by the rows of the closest rank-k matrix to A in the Frobenius norm. Projection is performed by computing the largest several hundred singular values and their corresponding left singular vector of A, so LSI is not scalable to massive databases. The scalability issue is resolved by the COV algorithm, which projects the problem into the subspace spanned by the k largest principal components of the symmetric, positive semi-definite attribute-attribute covariance matrix C:

$$C \equiv \frac{1}{M}\sum_{i=1}^{M} d_i d_i^T - \bar{d}\,\bar{d}^T , \tag{1}$$

where d_i represents the i-th document vector and $\bar{d}$ is the component-wise average over the set of all document vectors

A second approach for tackling the scalability issue with search systems is to identify sets of documents that cover similar topics, known as *clusters*, so they can be retrieved together to reduce the query response time. Cluster analysis can also be used to understand topics addressed by documents in massive databases. Implementation studies show that LSI and COV can successfully find *major* document clusters [5]. However, both algorithms are not as successful at finding smaller, *minor* document clusters, because major clusters dominate the process. During the dimensional reduction step in LSI and COV, documents in minor clusters are often mistaken for noise and are discarded.

Recently, two algorithms for identifying (possibly overlapping) multiple major and minor document clusters were reported [5]. The first, which is based on LSI, can only be applied to small databases. The second, which is based on COV, is scalable to large databases. The idea in both algorithms is to prevent major themes from dominating basis vector selection (for subspace projection) by introducing weighting. The weighting (or negative bias) decreases the relative importance of attributes represented by subspace basis vectors that have already been computed. The weighting is dynamically controlled to prevent deletion of information on major clusters. Both algorithms are refinements of a simpler one by Ando [1] proposed for a very small set of 683 TREC documents.

LSI-rescale (minor cluster mining based on LSI and re-scaling)
for $(i = 1; i \le k; i++)\{$
  $t_{\max} = \max(\ |r_1|,\ |r_2|,\ \ldots,\ |r_M|\ )$;
  $q = \text{func}\ (t_{\max})$;
  $R_s = [\ |r_1|^q\, r_1,\ |r_2|^q\, r_2,\ \ldots,\ |r_M|^q\, r_M\ ]^T$;
  SVD (R_s) ; (the singular value decomposition)
  $b_i' =$ the first row vector of V^T ;
  $b_i = \text{MGS}\ (b_i')$; (modified Gram-Schmidt)
  $R = R - R\, b_i b_i^T$; (residual matrix)

The input parameters for these algorithms are the document-attribute matrix A, the re-scaling factor q (used for weighting), and the dimension k to which the problem will be reduced. The *residual matrices* are denoted by R and R_s. Initially, R is set to be A. After each iterative step the residual vectors are updated to take into account the most recently computed basis vector b_i. After the k-th basis vector is computed, each document vector d_j in the original problem is mapped to its counterpart $\hat{d}_j$ in the k-dimensional subspace: $\hat{d}_j = [b_1, b_2, \ldots, b_k]^T d_j$. The query vector is mapped to the k-dimensional subspace before relevance ranking is performed.

The LSI-rescale algorithm is based on the idea that re-scaling document vectors after computation of each basis vector can be useful, but the weighting factor should be re-evaluated after each iterative step to take into account the length of the residual vectors to prevent decimation from over-reduction. More specifically, in the first step of the iteration, we compute the maximum length of the residual vectors and use it to define the scaling factor q that appears in the second step.

$$q = \begin{cases} t_{\max}^{-1} & \text{if } t_{\max} > 1 \\ 1 + t_{\max} & \text{if } t_{\max} \approx 1 \\ 10^{t_{\max}^{-2}} & \text{if } t_{\max} < 1 \end{cases}$$

The second algorithm, COV-rescale, for minor cluster identification is a modification of COV, analogous to LSI-rescale and LSI. In COV-rescale, the residual of the covariance matrix (defined by equation 1) is computed. Our implementation studies indicate that COV-rescale is better than LSI, COV, and LSI-rescale at identifying large and multiple minor clusters. In Section 5, we introduce *selective scaling*, a further improvement upon the weighting process in the LSI-rescale and COV-rescale algorithms.

COV-rescale (minor cluster mining based on re-scaling & COV)
for $(i = 1; i \leq k; i++)\{$
  $t_{\max} = \max(\ |r_1|,\ |r_2|,\ \ldots,\ |r_M|)$;
  $q = \text{func}\ (t_{\max})$;
  $R_s = [|r_1|^q\ r_1,\ |r_2|^q\ r_2,\ \ldots,\ |r_M|^q\ r_M]^T$;
  C = COV (R_s) ; (covariance matrix)
  SVD (C) ; (singular value decomposition)
  b_i' = the first row vector of V^T ;
  b_i = MGS (b_i') ; (modified Gram-Schmidt)
  $R = R - R\ b_i b_i^T$; (residual matrix)
$\}$

3. Our Mining System

Our prototype system consists of search and clustering engines based on VSM, PCA, and random sampling. Since it is uses VSM, with the exception of keyword labeling, its search and clustering engines can be applied to heterogeneous document sets (e.g., multilingual text, image audio, video), as long as the characteristics for the attributes are well defined. Notable features include its ability to mine and automatically label both major and minor clusters and to display

global and local views of the results from cluster mining. We introduce several new technologies that we have developed since our earlier report on a more primitive version of our current system [5]; these include: a refinement of the cluster identification algorithms to facilitate more efficient mining of small clusters; introduction of automatic cluster labeling; random sampling of documents to increase scalability of cluster mining; and addition of an advanced, user-friendly GUI. The GUI features: charts of retrieved documents and their relevancy rankings; multidimensional slices of attribute space, including a system to recommend slice angles based on the user's input query terms; an interface for browsing the collection of all retrieved clusters; options for selecting a subset of the collection and displaying only those clusters and topics covered by documents in those clusters; and keyword labels of retrieved clusters and titles of documents in clusters.

3a. Selective Scaling and Random Sampling for Basis Vector Computation

Our system uses the basic COV algorithm to mine major clusters. COV-rescale was developed to overcome difficulties with mining minor clusters using COV [5]. In this subsection we introduce *selective scaling (SS)*, a more computationally efficient minor cluster mining algorithm. Like COV-rescale, COV-SS allows users to skip over iterations that find major clusters and jump to iterations to find minor clusters. However, COV-rescale requires too many computations during the re-scaling step to be practical for massive databases. Selective re-scaling reduces computational costs by testing whether re-scaling is necessary after each step and performing the procedure only when necessary. Although the savings is measurable, it is not enough to enable mining from massive databases, so we combine COV-SS with random sampling to increase its scalability by several orders of magnitude. To summarize, our system COV is used to find major clusters, then COV-SS is used to find minor clusters for large databases, and COV-SS with random sampling for massive databases.

When random sampling of documents is used, concerns about the likelihood of selecting unrepresentative samples are often raised in a negative context. This concern is justified for our algorithm for extremely skewed, unrepresentative samples, such as sampling from only one or just a few clusters or sampling only noise. However, drawing a series of slightly unrepresentative samples is the most likely outcome of random sampling, and this lies at the heart of the success of our algorithm. Analysis of slightly unrepresentative samples tends to lead to better identification results since some minor clusters will appear to be larger than their actual size (when they are over-represented in the sample). Repetition of the sampling process increases the chance that all minor clusters will be over-represented, and consequently identified, at least once. This proposed algorithm belongs to class of randomized algorithms which output many of the clusters with appropriate labels *most* of the time, however, the probability of arriving at an incomplete set of clusters or misclassified (either as minor, medium or major) or mislabeled (keyword labels are misleading) exists regardless of the number and frequency of samples are drawn since a series of unrepresentative samples might be drawn from the large pool of data.

The input parameters for COV-SS are denoted as follows. $\mathbf{A}$ and k are defined as above, ρ is a threshold parameter, and μ is the scaling off-set. Initially, the residual matrix $\mathbf{R}$ is set to be $\mathbf{A}$. R does not be kept in the main memory; it suffices to keep just the the N-dimensional residual

vector $\mathbf{r}_i$ during each of the M loops. The output is the set of basis vectors $\{\mathbf{b}_i : i = 1, 2, \ldots, k\}$ for the k-dimensional subspace. M is either the total number of documents in the database or the number of randomly sampled documents from a very large database.

```
COV-SS  (A, k, ρ, μ, b )
for (int h = 1, h ≤ k, h++) {
   if (! first) for (int i = 1, i ≤ M, i++) {
      t = |r_i|;                                   (length of document vector)
      if (|| P[i] || > ρ ) {                       (dot product greater than threshold)
         w = (1 − ||P[i] ||)^(t+μ) ;               (compute scaling factor)
         r_i = r_i w ;                             (selective scaling)
         continue;
         }
      }
   C = (1/M) Σ_{i=1}^{M} r_i r_i^T  −  r̄ r̄^T ;     (compute covariance matrix)
   b_h  =  PowerMethod(C) ;                        (compute λ_max and its eigenvector)
   b_h  =  MGS(b_h) ;                              (Modified Gram-Schmidt)
   for (int i = 1, i ≤ M, i++) {
      Q[i] = r_i · b_h ; P[i] = ||r_i||^2 ;
      P[i] = Q[i] / √P[i] ;                        (store dot product = similarity measure)
      }
   for (int i = 1, i ≤ M, i++) ; r_i = r_i - Q[i] b_h ;   (residual matrix)
   if (first) first = false ;
   }
```

Here: $\mathbf{P}$ and $\mathbf{Q}$ are M-dimensional vectors; $\mathbf{R}$ is the residual matrix (which exists in theory, but is not allocated, in practice); $\mathbf{r}_i$ is the i-th document vector of $\mathbf{R}$ (an N-dimensional vector); $\bar{\mathbf{r}}$ is the component-wise average of the set of all residual vectors $\mathbf{r}$, i.e., $\bar{\mathbf{r}} = (1/M) \sum_{i=1}^{M} \bar{\mathbf{r}}$; $\mathbf{C}$ is the N-by-N square covariance matrix; w and t are double-precision floating point numbers; and *first* is a boolean expression equivalent to *true*.

COV-SS selectively scales document (residual) vectors according to the most recently computed similarity measure P[i] (i.e., the dot product of the previous basis vector and the document vector). The user-specified threshold and offset parameters control the number of minor clusters that will be associated with each basis vector. A small threshold and large offset value tend to lead to basis vectors associated with a large number of minor clusters. Conversely, a large threshold and a small offset value tend to lead to basis vectors with few minor clusters.

The computational work associated with the COV-based re-scaling algorithms (COV-rescale and COV-SS) is significantly greater than the basic COV algorithm. Re-scaling is computationally expensive for large databases. COV has no rescaling costs, but it uses a moderately expensive eigenvalue-eigenvector solver for large, symmetric positive semi-definite matrices. COV-rescale and COV-SS use the accurate and inexpensive *power method* to find the largest eigenvalue and its corresponding eigenvector after each round of (possible) re-scaling of residual vectors. An

analogous selective scaling algorithm for LSI (denoted by LSI-SS) can be constructed straightforwardly [6]. The computational trade-offs are similar for LSI with and without re-scaling. LSI requires no re-scaling of residual vectors, however, a moderately expensive solver (such as the Lanczos algorithm) must be used to determine left singular vector-value pairs of a very large, sparse document-attribute matrix. Like its counterpart COV-based algorithm, LSI-SS uses most of its computational resources to re-scale some residual vectors after finding each new singular value-vector pair, and it can partially off-set the additional cost by using the power method. LSI cannot be applied to massive databases without specialized hardware since it requires storage of all non-zero entries of document vectors. LSI-SS is even less scalable, because the system must cope with a dense matrix with the same dimension as the original document-attribute matrix after only a few steps; the original document-attribute matrix is typically only 0.2% to 0.3% dense.

3b. Cluster Mining: Finding and Labeling Clusters

Our system uses information from basis vector computations to find and label clusters using an embodiment of the *Label Algorithm* given below. The input consists of: basis vectors $\mathbf{b}_i$ output by COV or COV-SS; the number of cluster labels for extrinsic and intrinsic keywords (defined below); a threshold for separating clusters; and keyword data extracted from the original document data. We implemented the *Label Algorithm* as two separate two sub-programs: *cluster keyword generation* and *cluster merging and labeling.*

The cluster keyword generation program computes the similarity measure between each basis vector and all the document vectors in the original document-keyword matrix. It produces extrinsic and intrinsic keywords if the similarity measure between a basis vector and document vector is greater than a given threshold δ. *Extrinsic* keywords are the top $p(e)$ keywords that contribute towards making the similarity measure between a basis vector and the document vector greater than the threshold δ. *Intrinsic* keywords are the top $p(i)$ keywords that correspond to the largest $p(i)$ TF-IDF weights of the original document vector. Note that, unlike k-means and k-menoid clustering, our clustering method is not partition-based. In other words, we allow a document vector to be classified into more than one cluster as long as the extrinsic keywords of the document vector for two basis vectors have no keywords in common. Since database contents may be mined from different user perspectives allowing clusters to overlap is essential. Furthermore, information on overlaps between clusters can yield valuable information on relationships between clusters and topics in the documents.

Label Algorithm: Cluster detection and labeling

Input: $\{ \mathbf{b}_i : i = 1, 2, \ldots, k\}$ basis vectors

output: $\{s_{i,j} : j = 1, 2, \ldots, q_i\}$ clusters associated with $\mathbf{b}_i$,

where:

$s_{i,j} = \{\langle\text{cluster label}\rangle\langle\text{cluster type}\rangle\langle\text{document ID list}\rangle\}_{i,j}$,

q_i = number of clusters associated with $\mathbf{b}_i$,

$\langle$ cluster label $\rangle$ = keyword set $p(i) \bigcup p(e)$,

⟨ cluster union ⟩ = *major, minor* or *noise*, and
⟨ document ID list ⟩ = list of IDs for documents in the cluster.

4. Cluster Analysis Studies

We conducted numerical experiments with LSI, COV, COV-rescale and Cov-SS using a *Reuters* and *LA Times* TREC news databases with 21,578 and 127,742 articles, respectively. In our experiments, the LSI and COV algorithms found major clusters, but they usually failed to find all minor clusters. The algorithms deleted information in some minor clusters, because major clusters and their large sub-clusters dominated the subjects that were preserved during dimensional reduction. For medium size databases, major clusters should be mined using basic COV and minor clusters using COV with selective re-scaling.

Table 1: Comparison of LSI- and COV-based cluster analysis algorithms: scalabilities, bottlenecks, ability to identify major and minor clusters.

Algorithm		Scalability		Clusters found	
	variations	DB size	bottleneck	major	minor
LSI	basic	medium	main memory	++++	+
	Ando [1]	small	dense matrix	++	++
	+ re-scale	small	dense matrix	++++	+++
	+ s-scale*	small	dense matrix	++++	+++
	+ sampling	large	many samples	+	+++
	+ re-scale + samp	medium	dense matrix	+	+++
	+ s-scale + samp	medium	dense matrix	+	+++
COV	basic	large	no. attributes	++++	+
	+ re-scale	medium	re-scaling	++++	+++
	+ s-scale*	medium	re-scaling	++++	+++
	+ sampling	large	many samples	+	+++
	+ re-scale + samp	large	many samples	+	+++
	+ s-scale + samp	large	many samples	+	+++

for clusters found: ++++ = most , +++ = many , ++ = some , + = few

Major cluster identification results from LSI and COV are usually identical, however, COV usually requires 20%-30% fewer iterations to find major clusters because it can find clusters in both the negative and positive directions along each basis vector since the origin is shifted during dimensional reduction. The shift parameter is the second term in the LHS of equation (1). LSI can only find clusters either in the positive direction or in the negative direction of each basis vector, but not both. For massive databases, major clusters should be mined using basic COV , followed by COV with selective scaling and sampling to find minor clusters. Selective scaling is preferable to (complete) re-scaling since the results from both should be similar, but selective scaling is more computationally efficient.

Figures 1 and 2 and show results from our cluster identification experiments with basic COV and its variations and the *LA Times* database. 2000 documents were sampled each time during

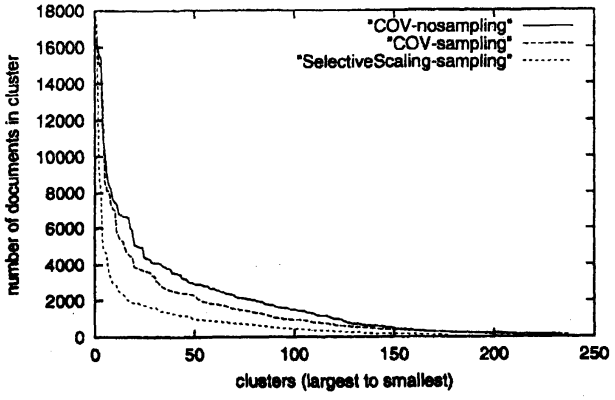

Figure 1: *Graph of size and number clusters identified by 3 mining methods: COV, COV using sampling, and COV using sampling & selective scaling.*

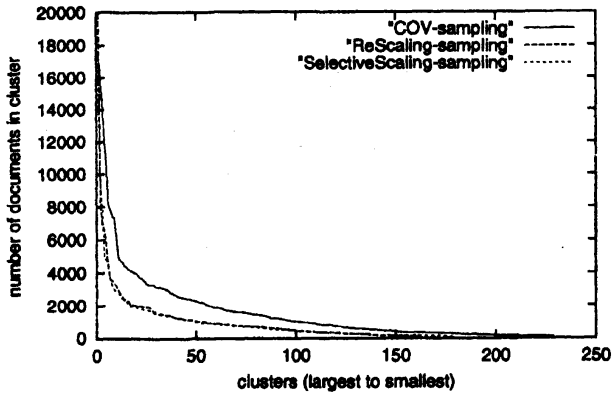

Figure 2: *Graph of size and number of clusters identified by 3 mining methods: COV using sampling, sampling & complete re-scaling, and sampling & selective scaling.*

a total of 3 random samplings. Identified clusters are sorted according to size to determine algorithms that are effective. The results show that both re-scaling and sampling skip over major clusters and more quickly discover minor clusters. Figure 2 confirms that results from complete and selective scaling are similar so that the more computationally efficient selective scaling should be used in practice. Table 1 summarizes the strengths and limitations of the algorithms.

Figure 3 shows a screen image from our system. The coordinate axes for the subspace are the first three basis vectors output by the COV algorithm. The 3 major clusters **A**, **B** and **C** shown are comprised of articles about {*Bush, US, Soviet, Iraq*}, {*team, coach, league, inning*}, and {*police, digest, county, officer*}, respectively. A trace of minor cluster **D** on {*Zurich, Swiss, London, Switzerland*} can be seen in the background. This minor cluster can be seen more clearly when other coordinate axes are used for visualization and display. This example shows that careful analysis is needed before deciding whether a faint cloud of points is noise or

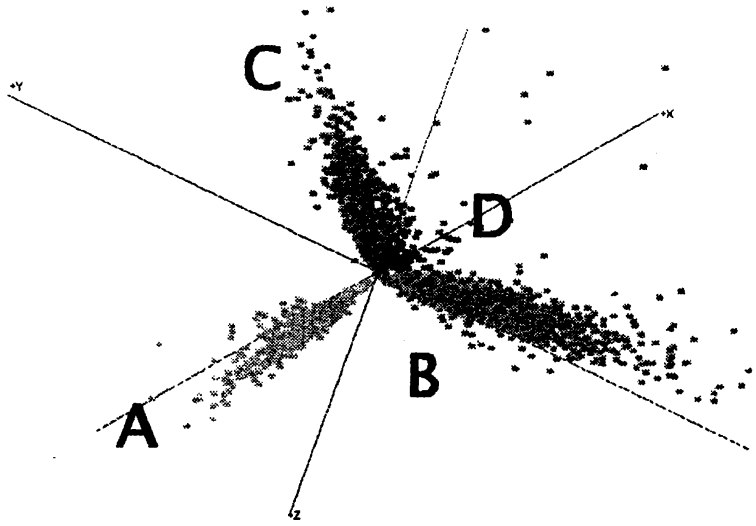

Figure 3: *Identification, labeling, and display of major clusters in the* LA Times *database.*

part of a cluster. Examples of minor clusters identified by our system are shown in Figure 4, where the basis vectors are 58, 63 and 104. Note that two clusters may lie along a coordinate axis - one each in the positive and negative directions. The clusters are comprised of articles on {*abortion, anti-abortion, clinic, Roe*}, {*lottery, jackpot, California, ticket*}, {*AIDS, disease, virus, patient*}, {*gang, school, youth, murder*}, {*Cypress, Santiago, team, tournament*}, and {*jazz, pianist, festival, saxophonist*}, respectively. A plus or minus sign is used to mark the direction in which clusters lie.

5. Conclusions and Future Work

We mention a number of interesting questions and and open problems that have arisen in the course of developing our system. They include:

- *How can we determine the optimal reduced dimension k (the so-called "intrinsic dimension") of the attribute space ?*
- *How can we determine whether a massive database is suitable for random sampling ?* Can we develop simple tests to determine whether there is an abundance of witnesses or if the database consists entirely of noise or documents on completely unrelated topics ?
- *How can one devise a reliable means for estimating the optimal sampling size for a given database ?* Factors to consider are the cluster structure of the database (the number and sizes of the clusters and the amount of noise) and the trade-off between sample sizes and the number of samples. Are there any advantages to be gained from dynamically changing the sample sizes based on clusters that have already been identified ?
- *What is a good stopping criterion for sampling ?* When is it appropriate for a user to decide that it is *likely* that *most* of the major or minor clusters have been found ?

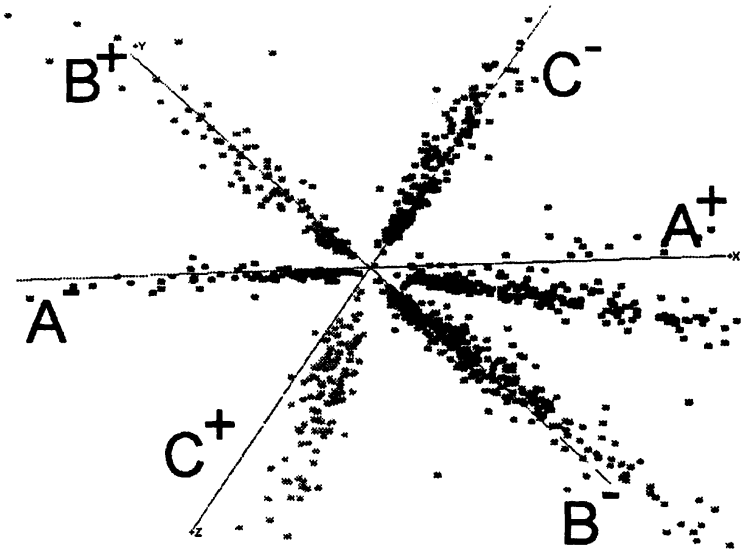

Figure 4: *Identification, labeling, and display of minor clusters in the* LA Times *database.*

- *How can the GUI effectively map identified clusters and their inter-relationships (subclusters of larger clusters, cluster overlap, etc.) ?*

Although many open issues and possibilities for improvements remain, our prototype system demonstrates that the COV algorithms with selective scaling and random sampling can be effective for exploring contents of very large databases.

Acknowledgements: The authors would like to thank E. Brown for providing access to TREC data and M. Houle, H. Samukawa and K. Takeda for helpful conversations.

[1] R. Ando, Latent semantic space, *Proc. of ACM SIGIR*, Athens, Greece, 213–223, July 2000.

[2] M. Berry, S. Dumais, G. O'Brien, Using linear algebra for intelligent information retrieval, *SIAM Review*, **37**(4), 571–595, Dec. 1995.

[3] S. Deerwester et al., Indexing by latent semantic analysis, *Journal of the American Society for Information Science*, **41**(6), 391–407, 1990.

[4] M. Houle, *private communication*, June 2002, (manuscript submitted for publication).

[5] M. Kobayashi, M. Aono, Major and outlier cluster analysis using dynamic re-scaling of document vectors, *Proc. of SIAM Text Mining Workshop*, Arlington, VA, 103–113, 2002.

[6] M. Kobayashi, M. Aono, Major and minor cluster mining using selective re-scaling and random sampling, *IBM Research Report*, RT-0491, July 1, 2002, de-classified Nov. 12, 2002.

[7] M. Kobayashi, L. Malassis, H. Samukawa, Retrieval and ranking of documents from a database, *patent*, filed June 2000 in Japan, March 14, 2002 in U.S..

[8] Sakano, K. Yamada, Horror story: the curse of dimensionality, *Information Processing Society of Japan (IPSJ) Magazine*, **43**(5), 562–567, May 2002 (in Japanese).

[9] G. Salton (ed.), *The Smart Retrieval System*, Prentice-Hall, Englewood Cliffs, NJ, 1971.